Building a Flexible Surface Characterisation System Architecture

X. Lan, X. Jiang, L. Blunt and S. Xiao
Centre for Precision Technologies
School of Computing and Engineering
University of Huddersfield, Queensgate, Huddersfield HD1 3DH, UK

ABSTRACT

Surface characterisation system needs to be updated constantly with the emergence of new algorithms and methods in the field of surface metrology. As function modules in current surface characterisation system are tightly coupled together, it is not conducive to the reuse of function modules and innovation of overall system. A lot of redundant and duplicate works have been done in present characterisation systems, and they will be done again when building a new characterisation system. In order to improve the reusability of function modules and facilitate system extension, this paper presents a flexible architecture for such surface characterisation systems by employing component based development technologies. Function components will be separated from system framework and implemented independently, so that the overall system is constructed by gluing these function components together.

Keywords Surface characterisation, Components, Flexible, Reuse

1 Introduction

The study of surface metrology is becoming more and more commonplace in industrial and research environments(Conroy and Armstrong 2005). Surface verification is based on the raw data which is acquired by appropriate instruments. At present, instruments companies or institutes are developing their own surface characterisation systems individually. As there may be something improper that needs to be replaced or new methods or algorithms added, the corresponding code for the implementation must be modified or appended to original application. It is not easy to find out where exactly to add the new code. Furthermore, the whole application solution would need to be recompiled and rebuilt and then distributed to the client to update. Hence, it is notoriously difficult to modify or extend the existing characterisation system.

Component Based Software Engineering(CBSE), which is a branch of software engineering , emphasizes the separation of concerns in respect of the wide-ranging functionality available throughout a given software system. Building software from components is not a new concept, and a typical way of designing complex software systems is always to start with identification of system parts designated subsystems or blocks, and modules, classes, procedures (on a lower level)(Thakur, Chen et al. 1999). However, all these system parts are always tightly coupled with each other, so the updating and maintaining of these system parts is greatly restricted by the whole system due to the lack of flexibility. The major goals of CBSE are the provision of support for the development of systems as assemblies of components, the development of components as reusable entities, and the maintenance and upgrading of systems by customising and replacing their components(Heineman and Councill 2001). It is means that one constructs the software system from pre components rather than “reinventing the wheel” each time.

2 Surface Characterisation System Architecture

The whole characterisation system is divided into one framework and a number of components. The framework, which should be a standalone executable platform, is an essential part of the system. It could be thought of as an empty system. Except the characterisation functions, all the other system functions should be realised in the framework. Although there are no real characterisation functions, this framework can invoke them though the pre-defined interfaces which are recognized as interactive protocols. Functional components are quite different from system framework. They only realise one

independent process method without taking relationships with other methods into account, while the framework mainly controls the analysis procedures rather than the concrete analysis methods. All these function components cannot run separately despite of that they are standalone executable blocks, and they need a container which refers to the system framework here to run.

In this characterisation system, function components can be classified as three types: data access components realise the data exchange between system and measurement output file; data process components analyse and process imported data; data display components show the processed data or analysis result for end user.

Since the system framework and functional components are implemented independently, it is necessary to specify their connective protocols which is named interface in component based software engineer. The interface is crucial to implement the dynamic connection of components rather than a statically chained function call, and it is not only the bridge that connects the components and framework; it also illustrates the functions, while the component is the function implementation. The connector is transparent to the framework which needs not to care about the implementation of components.

Each component in this characterisation system should implement one or more interfaces; otherwise it is unrecognisable for framework."SSObject" is the most basic and important interface (see figure 2), which should be implemented by every component in the system. It illustrates two functions: "OnRegister" and "UnRegister". The former is called as the component is added to the framework, while the latter is called as the component is removed from the framework. As long as one component implements this interface, it can be accepted as a system component. What functions that component can provide for clients (who use the component) depend on the interfaces which it has implemented.

3 Function Components

3.1 Data Access Components

Data access component implements system I/O functions. These components provide the raw data for the system and can acquire the measurement data from instruments in a direct or indirect way. If a data access component directly acquires measurement data from a particular instrument, it means that this component is a customized component which is usually bound to the instrument and cannot be used by other instruments. For the sake of good reusability, the indirect way to acquire measurement data is considered. The measurement data file is the bridge between instruments and data access components. As illustrated in figure 3, a measurement file is the input of data access component, while a data matrix is output for further analysis.

No matter which kind of techniques used to measure a surface, the result is usually stored in a file with specific format.ISO has specified the stand file format SDF(ISO25178-7 2009) for areal measurement data and SMD(ISO5436-2 2001) for profile data. However, not all instruments are compatible with the standard file format, and some instrument manufacturers create their own file format for data storing. For example, SUR is used as Mountains map surface format and Taylor Hobson surface format, while OPD is Wyko surface format. As there are so many kinds of file format for measurement data, it is impossible to parse all of them. Furthermore some new file formats may be utilised in the future.

3.2 Data process Components

Surface analysis and characterisation is a sequential procedure of several operations. Every operation in the chain of an operator, such as fitting, filtering and parameter calculation, can be encapsulated as a data process component. Some data manipulations such as data transformation and data arithmetic also belong to data process components. Data process components are a "black box" which is invisible for clients who use this component. As importing data to be processed, a data process component carry out certain operation on the input data and then output the processed data. What is the input and output data are much more critical than how to do the operation. According the I/O of data process components, these process components can be divided into three categories.

1) Two Data In and One Data Out (Figure 4)
2) One Data In and One Data Out (Figure 5)

3) One Data In and One Display Control Out (Figure 6)

3.3 Data Display Components

Although the data display component seems to have no relationship with surface characterisation process, it is helpful to have an intuitive sense of surface data and explicit understanding of each analysis procedure. Both the input data and output data of each process component need to be displayed by certain display component. A variety of ways can be used to express a surface data. For example, a data table can list out all the height values of a surface from the quantitative aspect, while graphical view which often gives an end user a better idea of surface feature can supply a qualitative aesthesis.

A display component only has surface data as the input and it is essentially a display control. When the system framework need a display component to show a surface data, any one of these display components can be created and then be embeded to the system framework. An integral surface data usually includes a discrete data set which is a matrix storing the height values and data instructions which specify the intervals, offsets, units and so on. It is not easy to present these height values of a surface. One simple way is to plot these discrete points and connect the adjacent point, and form a grid graphic of the matrix data. This grid graphic cannot well display the surface features especially when it is be viewed from the top. Fortunately, three dimensional graphics display is no longer a barrier as the development of computer graphical technology. Both OpenGL(Hill and Kelley 2000; Wright and Lipchak 2004) and DirectX can provide a perfect view of real objects, and many implementation details have been omitted. Figure 3.7 is the interface of a 3D display component which is implemented by utilising OpenGL technology and figure 3.8 affords a profile view of a surface data.

4 CONCLUSIONS

Surface characterisation technology is undergoing tremendous innovation and various creative methods are being employed to characterise surfaces. A traditional way to extend current characterisation system is to integrate new methods to it with system modifications. As a consequence, the whole system becomes increasingly large and complex(Crnkovic 2001). Components based development technology aims to construct software systems by gluing loosely coupled components. The complexity of whole system is determined by component types, rather than component quantities. In this characterisation system, file formats, analysis algorithms and display methods are three mutable system functions. Therefore, three kinds of components are classified to implement these functions respectively. These components are recognised as additional parts of whole system, and they can be added, substituted or removed without affecting other parts. Besides system developers, any end users also have the right to extend system functions. They can develop their own function components according to their specific requirement as long as such components have implemented pre-defined interfaces.

REFERENCES

Conroy, M. and J. Armstrong (2005). "A comparison of surface metrology techniques." Journal of Physics: Conference Series 13: 458-465.

Crnkovic, I. (2001). "Component-based software engineering-new challenges in software development." Software Focus 2(4): 127-133.

Heineman, G. T. and W. T. Councill (2001). Component-based software engineering: putting the pieces together, Addison-Wesley USA.

Hill, F. S. and S. M. Kelley (2000). Computer graphics: using OpenGL, Prentice Hall Upper Saddle River, NJ.

ISO5436-2 (2001). "Geometrical Product Specifications (GPS) - Surface texture: Profile method; Measurement standards- Part 2: Software measurement standards."

ISO25178-7 (2009). "Geometrical Product Specifications (GPS) - Surface texture: Areal - Part7: Software measurement standards."

Wright, R. S. and B. Lipchak (2004). OpenGL superbible, Sams Indianapolis, IN, USA.
Thakur, D. S., V. W. Chen, et al. (1999). "Systems Development Strategy: A Component-based Approach, The Prototype." Journal of the Association for Laboratory Automation 4(5): 28-33.

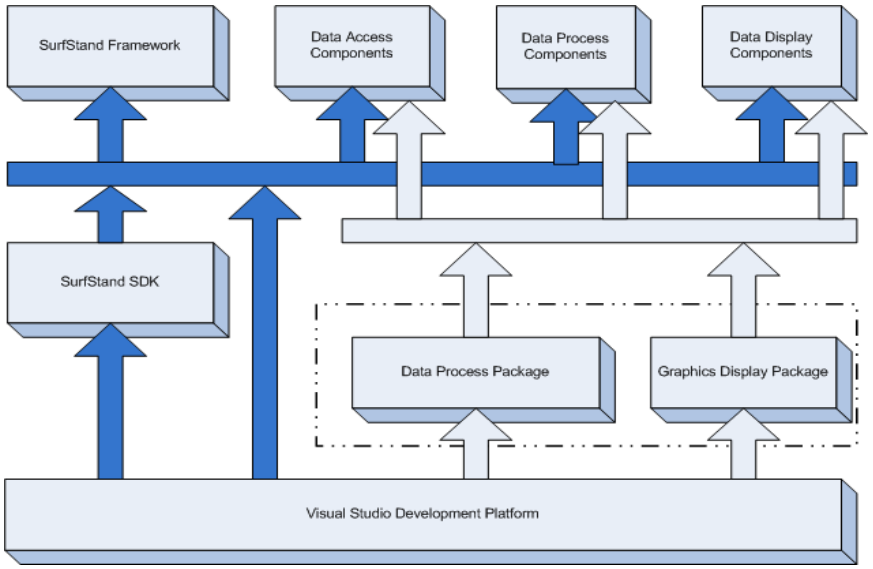

Figure 1: SurfStand" System Architecture Diagram

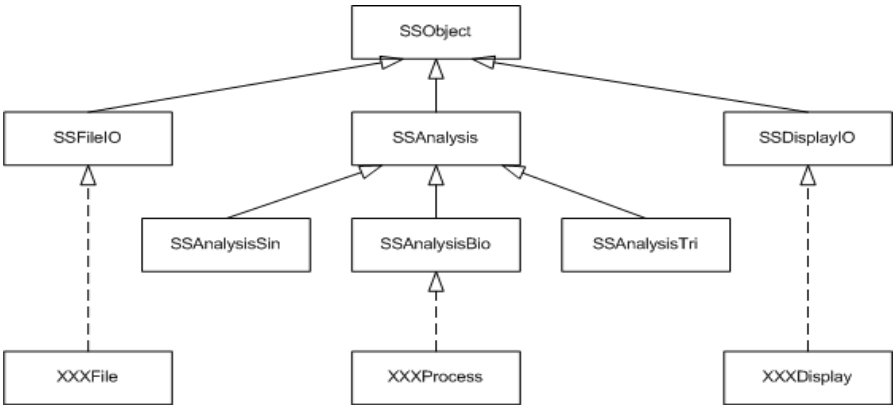

Figure 2: Component interfaces hierarchy diagram

Figure 3: The 4-sensor probe

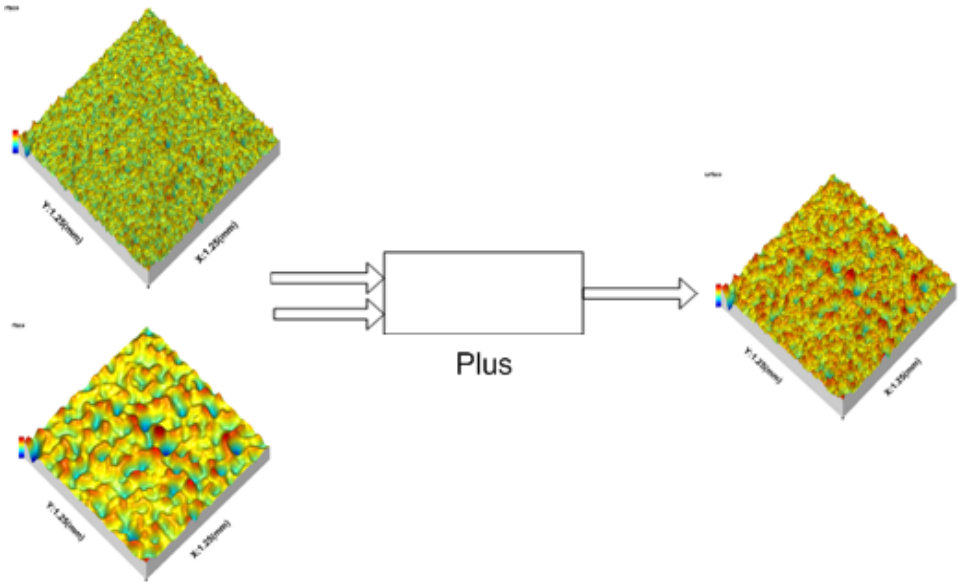

Figure 4: Example of data process type 1: plus two surfaces

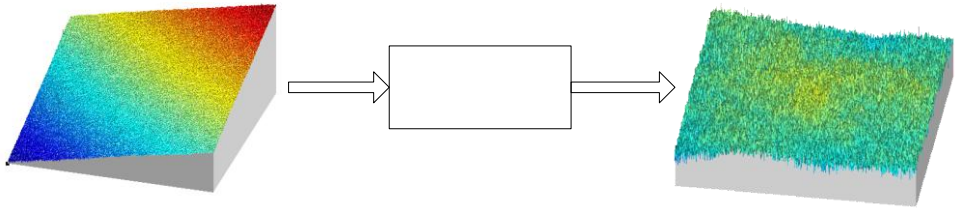

Figure 5: Example of data process type 2: levelling

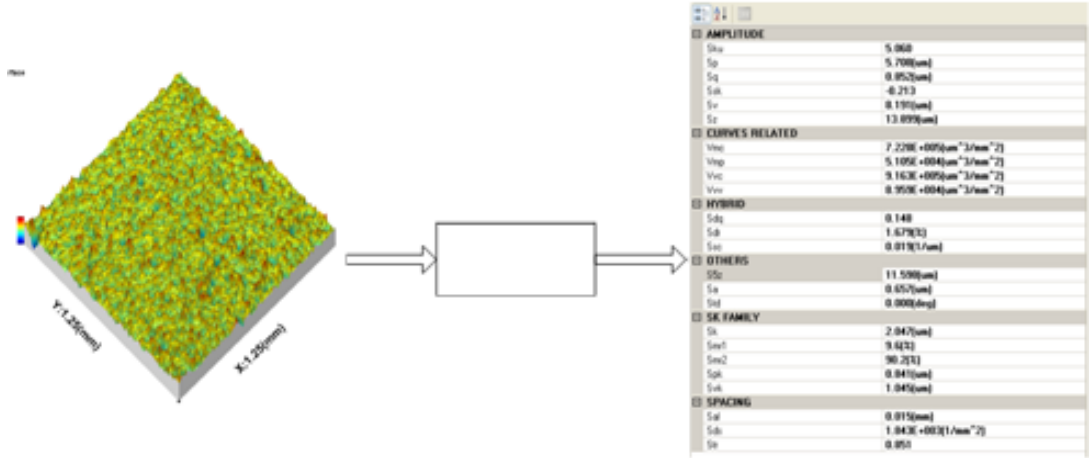

Figure 6 Example of data process type 3: parameters calculation